15-451/651 Algorithms, Fall 2017

Homework #7 **Due: December 6-8, 2017**

This is an oral presentation assignment. Again, there are three regular problems (#1-#3) and one programming problem (#4). You should work in groups of three. The presentation sign-up sheet will be online soon (details on Piazza), and your group should sign up for a 45 minute slot. Each person in the group must be able to present every regular problem. The TA/Professor will select who presents which regular problem. You are not required to hand anything in at your presentation, but you may if you choose.

The programming problem will be submitted to autolab, similar to HW#1. It is due at 11:59pm on Thursday, December 7. You will not have to present anything orally for the programming problem. Please include a comment in your program explaining the algorithm you used. You can discuss the problem with your group-mates (and only your group-mates), but must write the program by yourself. Please do not copy.

1. **Bridge Flipping Revisited**

   You're given a rooted tree T of n nodes. The tree never changes. Among the edges from a node to its children, exactly one of them is *solid*. The others are *dashed*. The choice of which child is solid may change over time. A sequence of Access() operations are done, as described here:

   Access(x): The path from x to the root of the tree is followed. Whenever a dashed edge is encountered a cost of one is incurred by the algorithm.

   Before, during, or at end of this process the algorithm is allowed to change which is the solid edge from any node to its children. The cost of each switch is one.

   The *Greedy* implementation of the Access(x) algorithm switches all the dashed edges on the path to solid at the beginning of the access, so its cost is just one for each dashed edge on the path from x to the root.

   The *Omniscent* algorithm is smarter. It can make decisions about which edges to flip based on the future as well as past accesses.

   The *Static* algorithm must make its choices of which edges are to be dashed and which are to be solid at the very beginning of the access sequence, and then never changes these choices.

   (a) Assuming that Omniscent and Greedy both start with the same tree and same arrangement of solid and dashed edges, prove that Greedy is always at most a factor of 2 more expensive than Omniscent. (I.e. Greedy is 2-competitive.) Use a potential function.

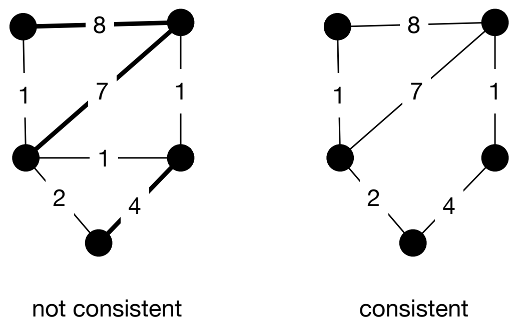
   (b) Give a strategy for Static such that any access costs it at most $\log_2 n$. Note that these two parts together show that Greedy costs at most $2 \log n$ amortized per access.

2. **Hardness of Being Consistent**

Consider the following problem:

> **Metric Consistency.** *Instance:* Let $G = (V, E)$ be an undirected graph (no self loops, no multiedges), with positive edge lengths $\ell(e)$ and a positive reward $r(e)$ for each edge. The reward of a set E' of edges is $\sum_{e \in E'} r(e)$, and the length of a path is the sum of its edge lengths. Let R be a positive number.
>
> *Question:* Is there a subgraph $G' = (V, E')$ with $E' \subseteq E$ such that for every edge $\{u, v\} \in E'$, the length of each path connecting u to v is $\geq \ell(\{u, v\})$ and the reward of E' is $\geq R$.

In other words, in the subgraph G' that the problem seeks, there are no "shortcuts": if it exists, the direct edge between a pair of vertices u, v is always at least as short as any other path. We call a graph where this is true for every edge *consistent.* The figure below shows a consistent subgraph and an inconsistent subgraph:



Numbers on the edges above represent edge lengths $\ell(e)$. On the left, **bold** edges have shortcuts.

Prove that this problem is NP-complete. You may use any NP-complete problem in the course lecture notes or discussed in class. The following fact may be useful:

Fact: A graph with all equal edge lengths is always consistent.

3. **DFS for Approximate Vertex Cover**

Recall the minimum vertex cover problem:

> **Minimum Vertex Cover.** Let $G = (V, E)$ be an undirected graph (no self loops, no multiedges). Find the smallest set $S \subseteq V$ such that for each edge $\{u, v\} \in E$ either $u \in S$ or $v \in S$ (or both).

Consider the following algorithm to find an approximately optimal vertex cover:

DFS-for-VC Algorithm:

1. Do a DFS in G. Let F be the resulting DFS spanning forest.

2. Return all the non-leaf vertices of F. (Call this set S)

Answer the following questions regarding this algorithm:

(a) Prove that the set S produced by the above algorithm is a vertex cover (i.e. every edge is covered by some vertex).

(b) Prove that S is no bigger than 2 times the smallest possible vertex cover. Hint: think about possible lower bounds on the size of the vertex cover.

(c) Give an example where the algorithm achieves as close to the worst possible performance (in terms of approximation factor) as you can.

4. **Graph Slope** (Programming Problem)

In this problem you will write a program that takes as input an undirected graph $G = (V, E)$, where each edge $e \in E$ has a positive integer edge length $\ell(e)$, and a subset of vertices have a (possibly negative) integer height $h(v)$. Your program will assign real valued heights to all the remaining vertices so as to minimize the maximum *slope* of all the edges.

Given an assignment of heights to all the vertices, the slope of an edge $e = (u, v)$ is defined as follows:

$$\text{slope}(e) = \frac{|h(u) - h(v)|}{\ell(e)}$$

Input:

The first line of input contains n and m the number of vertices and edges of G. $1 \leq n \leq 1000$, $1 \leq m \leq 2000$. The following n lines each contain either the character "$*$", or an integer. The $*$ indicates that the hight is unspecified, and is to be computed by your program.

The following m lines each contain a pair of vertex numbers in the range $[0, n-1]$, and an integer representing the length of that edge. There are no multi-edges or self-loops. The length will be an integer in the range $[1, 10000]$. There will be no edges connecting a pair of vertices with known heights.

Output:

Output the minimum obtainable slope with six digits after the decimal point. Your answer should be within a relative or absolute error of at most 10^{-6} of the correct answer. The time limit is 10 seconds.

Examples:

```
3 2                                          slope = 5.000000
0
*
-100
0 1 1
1 2 19
```

```
6 5                                          slope = 3.500000
*
8
1
2
5
4
0 1 1
0 2 1
0 3 1
0 4 1
0 5 1
```